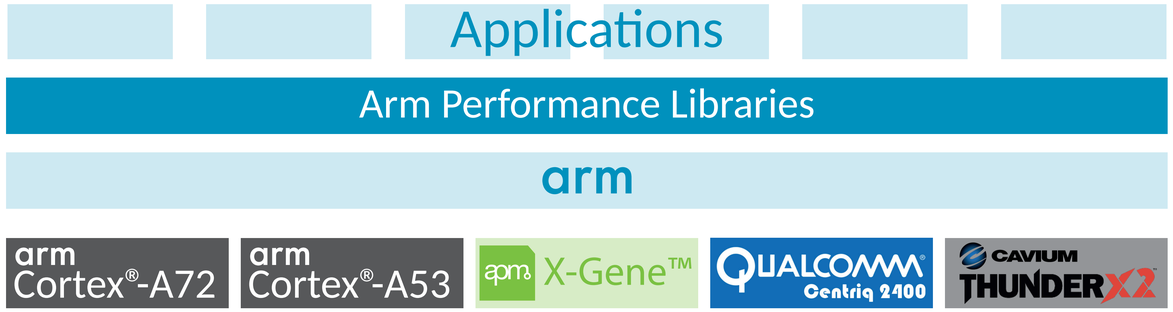
Performance portability across Arm microarchitectures

The Arm ISA provides **functional portability** across a range of microarchitectures from different Arm licensees.



Arm Performance Libraries provide the **performance portable** BLAS, LAPACK and FFT routines that the HPC ecosystem requires.

How do we quickly and effectively tune for all of these microarchitectures?

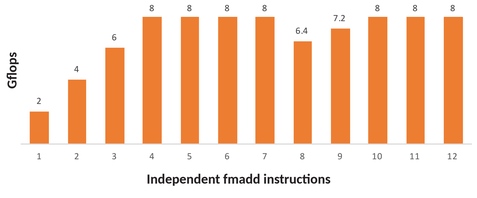
"Spotlight"

64-bit Arm Assembly Kernel Benchmark Reports

- What is the latency of a floating-point fused multiply-add (floating-point and vector)?
- What is the fastest way to load 64 bytes from memory?
- How many floating-point fused multiply-add instructions hide the latency of a load (for several different types of load)?
- What is the performance penalty for unaligned loads (for several different types of load)?
- Is it better to group load instructions together, or interleave with computation?
- What is the cost of moving data between registers (general-purpose, floating-point and vector)?
- What is the pipeline disruption caused by prefetching?

Fused multiply-add latency?

How many independent scalar FMADD instructions before we can reuse destination registers without performance penalty?




```
fmadd d0, d30, d31, d0
fmadd d1, d30, d31, d1
fmadd d2, d30, d31, d2
fmadd d3, d30, d31, d3
fmadd d4, d30, d31, d4
fmadd d5, d30, d31, d5
fmadd d6, d30, d31, d6
fmadd d7, d30, d31, d7
```

arm Cortex®-A57 **4**

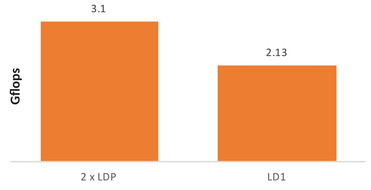
arm Cortex®-A53 **5**

apm X-Gene™ **3**

CAVIUM THUNDERX **6**

Fastest way to load 64 bytes (512 bits)?

Will two LDP instructions (each of which loads a pair of 128-bit vectors) outperform a single LD1 instruction (which loads four 128-bit vectors in one go)?



CAVIUM THUNDERX **Yes**

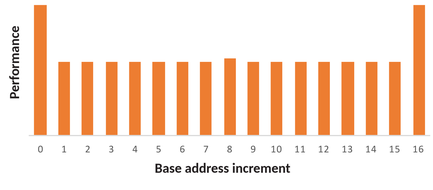
arm Cortex®-A53 **No**

apm X-Gene™ **No**

arm Cortex®-A57 **No**

Performance penalty for unaligned loads?

In a loop, perform 20 FMLA instructions, then load 128 consecutive bytes from memory. What is the effect if we increment the base address by 0-16 bytes each time around the loop?



CAVIUM THUNDERX **Yes: Prefers 16 byte alignment**

arm Cortex®-A53 **Yes: Prefers 8-byte alignment**

apm X-Gene™ **No**

arm Cortex®-A57 **No**

Acknowledgements: Chris Armstrong, Chris Goodyer and Joe Dobson from the Arm Performance Libraries team in Manchester, UK.

arm